

The Continuous 1.5D Terrain Guarding Problem: Discretization, Optimal Solutions, and PTAS*

Stephan Friedrichs,^{†‡} Michael Hemmer,[§] James King,[¶] and Christiane Schmidt^{||}

Abstract

In the NP-hard [33] continuous 1.5D Terrain Guarding Problem (TGP) we are given an x -monotone chain of line segments in \mathbb{R}^2 (the *terrain* T) and ask for the minimum number of guards (located anywhere on T) required to guard all of T . We construct guard candidate and witness sets $G, W \subset T$ of polynomial size such that any feasible (optimal) guard cover $G^* \subseteq G$ for W is also feasible (optimal) for the continuous TGP. This discretization allows us to (1) settle NP-completeness for the continuous TGP, (2) provide a Polynomial Time Approximation Scheme (PTAS) for the continuous TGP using the PTAS for the discrete TGP by Gibson et al. [23], and (3) formulate the continuous TGP as an Integer Linear Program (IP). Furthermore, we propose several filtering techniques reducing the size of our discretization, allowing us to devise an efficient IP-based algorithm that reliably provides optimal guard placements for terrains with up to 10^6 vertices within minutes on a standard desktop computer.

1 Introduction

In the 1.5D Terrain Guarding Problem (TGP), we are given an x -monotone chain of line segments in \mathbb{R}^2 , the terrain T , on which we have to place a minimum number of point-shaped guards, such that they cover T . This is a close relative of the Art Gallery Problem (AGP) and traditionally motivated by the optimal placement of antennas for line-of-sight communication networks, or the placement of street lights or security cameras along roads [1].

The authors would like to revive a motivation stemming from research regarding algorithms solving the AGP [10, 12, 16, 17, 34] already mentioned in [1]: An application of the AGP is the placement of sensors or communication devices w.r.t. obstacles, for example placing laser scanners in production facilities to acquire a precise mapping of the facility [16, 34]. While the AGP properly models most indoor environments it cannot capture many outdoor scenarios, like placing cell phone towers in an urban environment, because

*This work extends and subsumes Chapter 3 of *James King's PhD thesis*, pages 29–72, 2010 [32], and the extended abstracts that appeared in the *Proceedings of the 26th Canadian Conference on Computational Geometry (CCCG 2014)*, pages 367–373, 2014 [19] and in the *31st European Workshop on Computational Geometry (EuroCG 2015)*, pages 212–215, 2015 [20].

[†]Max Planck Institute for Informatics, Saarbrücken, Germany, sfriedri@mpi-inf.mpg.de

[‡]Saarbrücken Graduate School of Computer Science

[§]TU Braunschweig, IBR, Algorithms Group, Braunschweig, Germany, mhsaar@gmail.com

[¶]D-Wave Systems, Burnaby, Canada, jking@dwavesys.com

^{||}Communications and Transport Systems, ITN, Linköping University, Sweden. Supported by grant 2014-03476 from Sweden's innovation agency VINNOVA. christiane.schmidt@liu.se

it does not take height information into account. To remedy this shortcoming essentially means working on two dimensions and height, a 2.5D AGP. One dimension and height, the 1.5D TGP, is a natural starting point to develop techniques for a 2.5D AGP. We show in this paper that the “height dimension” is more benevolent than the “second dimension” in the AGP: It allows a finite discretization whose existence in the AGP is, to the best of our knowledge, still unknown and poses a key challenge w.r.t. software solving the AGP [10]. We hope that our contribution helps tackling the 2.5D AGP.

1.1 Our Contribution

- (1) Our core contribution is to show that the Continuous Terrain Guarding Problem (CTGP), where guards can be freely placed on the terrain, has a discretization of polynomial size (Section 2). We then infer two results:
 - (a) While the CTGP is known to be NP-hard [33], we also conclude that it is a member of NP, and hence NP-complete (Section 3).
 - (b) It follows from the Polynomial Time Approximation Scheme (PTAS) for the discrete TGP from Gibson et al. [23] that there is a PTAS for the CTGP (Section 4).
- (2) We present filtering techniques reducing the size of our discretization (Section 5).
- (3) An efficient algorithm for continuous and discrete TGP versions is proposed. It finds optimal solutions for terrains with up to 10^6 vertices on a standard desktop computer¹ within minutes. This is achieved following the Exact Geometric Computation (EGC) paradigm, i.e., using exact arithmetic for geometric calculations to ensure correctness. We test our algorithm and filtering techniques (Sections 6 and 7).

1.2 Related Work

The TGP is closely related to the AGP where, given a polygon P , we seek a minimum cardinality guard set that covers P . Potential guards can, e.g., be located on the vertices only, on arbitrary points in P , or patrol along edges or diagonals of P . Many polygon classes have been considered for the AGP, including simple polygons, polygons with holes, and orthogonal polygons. Moreover, the guards’ task can be altered, e.g. Laurentini [35] required visibility coverage for the edges of P , but not the interior.

The first result in the context of the AGP was obtained by Chvátal [5] who proved the *Art Gallery Theorem*, answering a question posed by Victor Klee in 1973 (see [38]): $\lfloor \frac{n}{3} \rfloor$ guards are always sufficient and sometimes necessary to guard a polygon of n vertices. A simple and elegant proof of the sufficiency was later given by Fisk [18]. Related results were obtained for various polygon classes, Kahn et al. [27] established a tight bound of $\lfloor \frac{n}{4} \rfloor$ for orthogonal polygons with n vertices.

The work of Chvátal and its variants focused on upper bounds on the number of guards. However, the AGP also is an optimization problem (given a polygon, find a minimum number of guards covering it), the decision variant of which was shown to be NP-hard for

¹Standard as of 2015: An Intel Core i7-3770 CPU with 3.4 GHz with 14 GB of main memory.

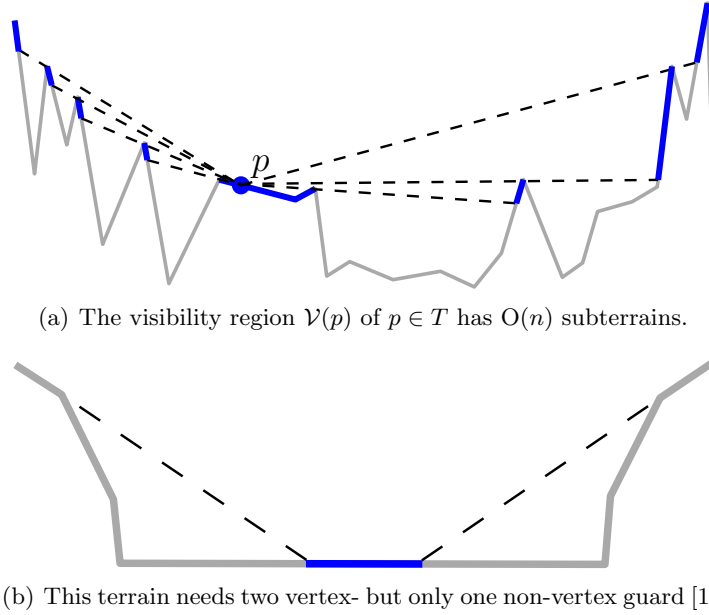


Figure 1: The Terrain Guarding Problem (TGP): visibility (a) and non-vertex guards (b).

various problem versions [39, 41], even for vertex guards in polygons without holes [36]. Eidenbenz et al. established APX-hardness of many AGP variants [14]. Chwa et al. [6] considered witnessable polygons, in which coverage of some finite set of witness points implies coverage of the entire polygon. For classical surveys on the AGP see O’Rourke [38] and Shermer [42], and de Rezende et al. [10] for more recent computational developments.

For the 1.5D TGP, research first focused on approximation algorithms, because NP-hardness was generally assumed, but had not been established. The first constant-factor approximation was given by Ben-Moshe et al. [1] for the discrete vertex guard problem version $\text{TGP}(V, V)$,² where only vertex guards are used to cover only the vertices. They were able to use it as a building block for an $O(1)$ -approximation of $\text{TGP}(T, T)$, where guards on arbitrary locations on T must guard all of T . The approximation factor of this algorithm was not stated by the authors, but claimed to be 6 in [31] (with minor modifications). Another constant-factor approximation based on ϵ -nets and Set Cover (SC) was given by Clarkson and Varadarajan [7]. King [31] presented a 4-approximation (which was later shown to actually be a 5-approximation [30]) for $\text{TGP}(V, V)$ and $\text{TGP}(T, T)$. The most recent $O(1)$ -approximation was presented by Elbassioni et al. [15]: Using LP-rounding techniques, they achieve a 4-approximation of $\text{TGP}(T, T)$ and $\text{TGP}(G, W)$ w.r.t. finite, disjoint $G, W \subset T$ (a 5-approximation if $G \cap W \neq \emptyset$). This approximation is also applicable to the TGP with weighted guards. In the 2009 conference version of [23], Gibson et al. devised a PTAS based on local search for $\text{TGP}(G, W)$ and $\text{TGP}(G, T)$, where $G, W \subset T$ are finite.

Only after all these approximation results, in the 2010 conference version of [33], King and Krohn established the NP-hardness of both the discrete and the continuous TGP by a reduction from PLANAR 3SAT. The membership of the CTGP in NP remained, to the best

² $\text{TGP}(G, W)$ means that W must be covered using only guards located in G , see Definition 1.1.

of our knowledge, an open problem that we answer positively in Section 3. Khodakarami et al. [29] showed that the TGP is fixed-parameter tractable w.r.t. the *depth of terrain onion peeling*, the number of layers of upper convex hulls induced by a terrain.

Variants of the TGP include guards hovering above the terrain (Eidenbenz [13]), orthogonal terrains (Katz and Roisman [28]), and directed visibility (Durocher et al. [11]). Hurtado et al. [26] gave algorithms for computing visibility regions in 1.5D and 2.5D terrains. Haas and Hemmer [25] presented implementations for 1.5D visibility based on [26] and the triangular expansion technique for visibility computations in polygons by Bungiu et al. [3].

Martinović et al. [37] proposed an approximate solver for the discrete TGP. Requiring a-priori knowledge about pairwise visibility of the vertices V , they 5.5- and 6-approximate $\text{TGP}(V, V)$ instances with up to 8000 vertices and dense (0.19–0.65) visibility matrices in 11–900 and 4–250 seconds. As geometric information is encoded in the input, they are not tied to the EGC paradigm, use floating-point arithmetic, and a parallel GPU implementation. Note that we solve a different problem: We determine the discretization and visibility information that Martinović et al. require as input, follow the EGC paradigm, and guarantee optimal solutions in no more than 3.5 seconds for 10000 vertices. However, we do not focus on dense visibility matrices and use different hardware, rendering a comparison of computation times meaningless.

Regarding a discretization for the continuous TGP, Gibson et al. claimed [23] that their local search works well, but could not limit the number of bits representing the guards. King gave a discretization with $O(n^3)$ guard candidates and $O(n^4)$ witnesses in his PhD thesis in 2010 [32] and posed the question if a smaller discretization exists. Independently, Friedrichs et al. discovered a discretization using $O(n^2)$ guard candidates and $O(n^3)$ witnesses [19] in 2014. This paper subsumes and extends [19, 32].

1.3 Preliminaries and Notation

A *terrain* T , see Figure 1, is an x -monotone chain of line segments in \mathbb{R}^2 defined by its *vertices* $V(T) = \{v_1, \dots, v_n\}$ that has *edges* $E(T) = \{e_1, \dots, e_{n-1}\}$ with $e_i = \overline{v_i v_{i+1}}$. Unless specified otherwise, $n := |V(T)|$. Where T is clear from context, we occasionally abbreviate $V(T)$ and $E(T)$ by V and E . v_i and v_{i+1} are the vertices of the edge e_i , and $\text{int}(e_i) := e_i \setminus \{v_i, v_{i+1}\}$ is its *interior*. Due to monotonicity, the points on T are totally ordered w.r.t. their x -coordinates. For $p, q \in T$, we write $p \leq q$ ($p < q$) if p is (strictly) left of q , i.e., has a (strictly) smaller x -coordinate. We refer to a closed, connected subset of T as a *subterrain*.

A point $p \in T$ *sees* or *covers* $q \in T$ if and only if \overline{pq} is nowhere below T . $\mathcal{V}(p)$ is the *visibility region* of p with $\mathcal{V}(p) := \{q \in T \mid p \text{ sees } q\}$. Observe that $\mathcal{V}(p)$ is not necessarily connected and is the union of $O(n)$ subterrains, see Figure 1(a). We say that $q \in \mathcal{V}(p)$ is *extremal* in $\mathcal{V}(p)$ if q has a maximal or minimal x -coordinate within its subterrain in $\mathcal{V}(p)$. For $G \subseteq T$ we abbreviate $\mathcal{V}(G) := \bigcup_{g \in G} \mathcal{V}(g)$. A set $G \subseteq T$ with $\mathcal{V}(G) = T$ is named a (*guard*) *cover* of T . In this context, $g \in G$ is sometimes referred to as *guard*.

Definition 1.1 (Terrain Guarding Problem). *In the Terrain Guarding Problem (TGP), abbreviated $\text{TGP}(G, W)$, we are given a terrain T , and sets of guard candidates and witnesses $G, W \subseteq T$. $C \subseteq G$ is feasible w.r.t. $\text{TGP}(G, W)$ if and only if $W \subseteq \mathcal{V}(C)$. If C is feasible and $|C| = \text{OPT}(G, W) := \min\{|C| \mid C \subseteq G \text{ is feasible w.r.t. } \text{TGP}(G, W)\}$, we say that C*

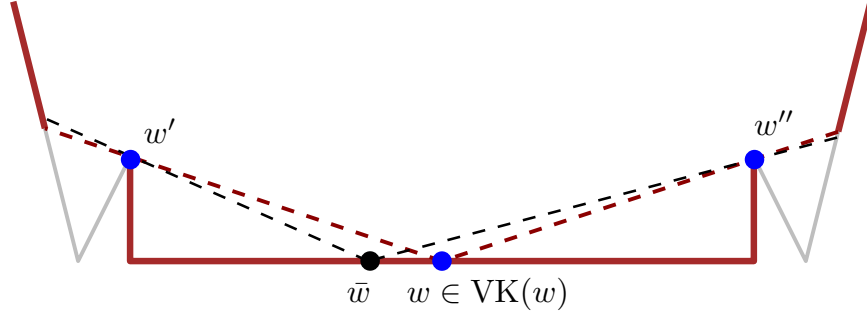


Figure 2: Witness w , $\mathcal{V}(w)$ highlighted in red, and its finite visibility kernel $\text{VK}(w) = \{w, w', w''\}$ marked in blue. \bar{w} has equivalent properties.

is optimal w.r.t. $\text{TGP}(G, W)$. $\text{TGP}(G, W)$ asks for an optimal guard cover $C \subseteq G$. The Continuous Terrain Guarding Problem (CTGP) is $\text{TGP}(T, T)$, and the Terrain Guarding Problem with Vertex Guards (VTGP) is $\text{TGP}(V(T), T)$.

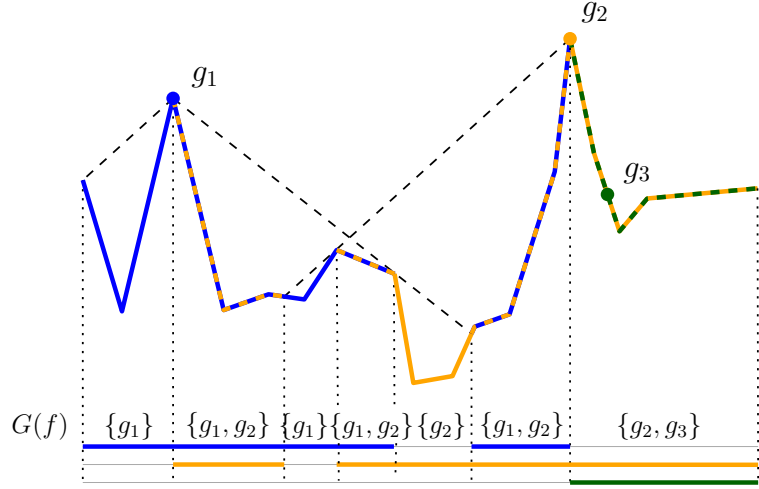
Throughout this paper, we assume $W \subseteq \mathcal{V}(G)$, i.e., that $\text{TGP}(G, W)$ has a feasible solution. The CTGP is the primary focus of this paper. Observe that CTGP and VTGP are different problems [1], as demonstrated in Figure 1(b). We consider VTGP a representative of the numerous discrete versions of the TGP; our algorithm solves both CTGP and VTGP, and generalizes to arbitrary discretizations.

2 Discretization

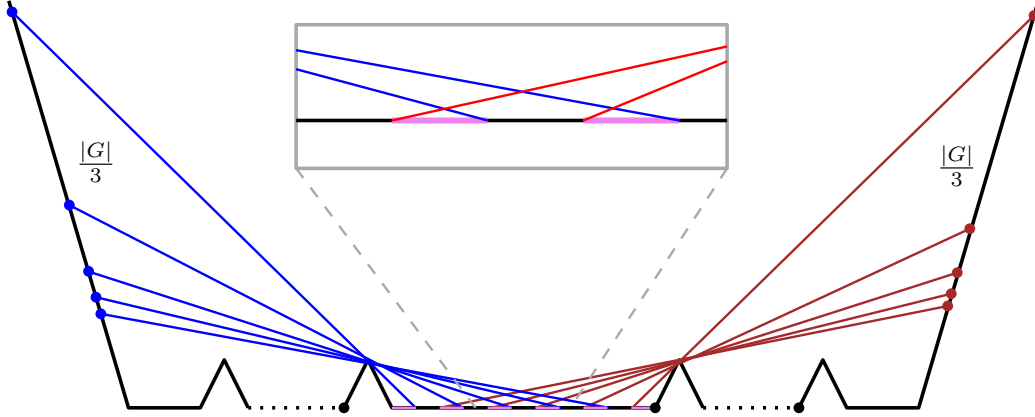
This section is our core contribution. We consider the following problem: Given a terrain T with n vertices, construct sets $G, W \subset T$ (guard candidate and witness points) of size polynomial in n , such that any feasible (optimal) solution for $\text{TGP}(G, W)$ is feasible (optimal) for $\text{TGP}(T, T)$ as well. We proceed in three steps. (1) In Section 2.1 we assume that we are provided with some finite guard candidate set $G \subset T$ and show how to construct a witness set $W(G)$ with $|W(G)| \in O(n|G|)$, such that any feasible solution of $\text{TGP}(G, W(G))$ is feasible for $\text{TGP}(G, T)$ as well. (2) Section 2.2 discusses a set of guard candidates U with $|U| \in O(n^2)$ and $\text{OPT}(U, T) = \text{OPT}(T, T)$. (3) We combine the above steps in Section 2.3.

When discretizing a problem as closely related to the Art Gallery Problem (AGP) as the Terrain Guarding Problem (TGP), one must consider the work of Chwa et al. who pursued the idea of *witnessable* polygons [6] which allow placing a finite set of witnesses, such that covering the witnesses with any guard set implies full coverage of the polygon. The basic building blocks of Chwa et al. are *visibility kernels*: Given a point w in a polygon, the visibility kernel of w is the set of points that see at least as much as w (definition for terrains below). Chwa et al. show that a polygon admits a finite witness set if and only if it can be covered by a finite set of visibility kernels; this is not the case for arbitrary polygons.

Transferring this approach to the TGP means that the visibility kernel of $w \in T$ is $\text{VK}(w) := \{w' \in T \mid \mathcal{V}(w) \subseteq \mathcal{V}(w')\}$. Then for the terrain T and $w \in T$ in Figure 2 we have $\text{VK}(w) = \{w, w', w''\}$, so $\text{VK}(w)$ is finite. The same argument holds for infinitely many $\bar{w} \in T$ near w . It follows that T does not admit a finite visibility kernel cover and thus is



(a) Visibility overlay of $\mathcal{V}(g_1)$, $\mathcal{V}(g_2)$, and $\mathcal{V}(g_3)$ indicated in blue, orange, and green, respectively. Overlaps are indicated by altering colors.



(b) The set of inclusion-minimal features may still have cardinality $O(n|G|)$.

Figure 3: Witness discretization: visibility overlay (a) and cardinality (b).

not witnessable as defined by Chwa et al. (our witness set below is different in that it is associated with a finite guard set).

2.1 Witnesses

Suppose we are given a terrain T and a finite set $G \subset T$ of guard candidates with $\mathcal{V}(G) = T$, and we want to cover T using only guards $C \subseteq G$, i.e., we want to solve $\text{TGP}(G, T)$. G could be the set $V(T)$ of vertices to solve the Terrain Guarding Problem with Vertex Guards (VTGP) or any other finite set, especially our guard candidates in Equation (5). We construct a finite set $W(G) \subset T$ of $O(n|G|)$ witness points, such that feasible solutions for $\text{TGP}(G, W(G))$ also are feasible for $\text{TGP}(G, T)$.

Let $g \in G$ be one of the guard candidates. $\mathcal{V}(g)$ subdivides T into $O(n)$ subterrains, see Figure 1(a). The monotonicity of T allows us to project them onto the x -axis and

thus to represent $\mathcal{V}(g)$ as a set of closed *visibility intervals*. We consider the overlay of all visibility intervals of all guard candidates in G , see Figure 3(a) for an overlay of three guard candidates. It forms a subdivision consisting of *maximal intervals* (maximal, connected intervals seen by the same guards) and *end points*. Every point in a *feature* f (maximal interval or end point) of the subdivision is seen by the same set of guards

$$G(f) := \{g \in G \mid f \subseteq \mathcal{V}(g)\}. \quad (1)$$

Observation 2.1. *Let f be a feature of the guard candidates' overlay, and let $g \in G$ be a guard. Now consider an arbitrary witness $w \in f$. Then*

$$w \in \mathcal{V}(g) \iff f \subseteq \mathcal{V}(g) \stackrel{(1)}{\iff} g \in G(f). \quad (2)$$

Observation 2.2. *Let G be a finite set of guard candidates and f a feature in the overlay of G . Then a witness point $w \in f$ can be represented by the set $G(f)$ of guards covering f .*

Placing one witness in every feature of the subdivision ensures that coverage of all these witnesses implies coverage of all features and thus of T . This requires $O(n|G|)$ witnesses. However, keeping efficient algorithms in mind, we reduce the number of witnesses, see Section 5.3: Similar to the shadow atomic visibility polygons in [8] — a successful strategy in AGP algorithms [10] — it suffices to include only those features f with inclusion-minimal $G(f)$, i.e., those for which no f' with $G(f') \subset G(f)$ exists:

Theorem 2.3. *Consider a terrain T and a finite set of guard candidates G with $\mathcal{V}(G) = T$. Let F_G denote the set of features of the visibility overlay of G on T and $w_f \in f$ an arbitrary point in the feature $f \in F_G$. Then for*

$$W(G) := \{w_f \mid f \in F_G, G(f) \text{ is inclusion-minimal}\}, \quad (3)$$

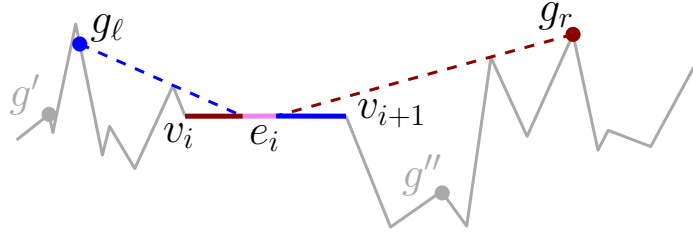
we have that if $C \subseteq G$ is feasible w.r.t. $\text{TGP}(G, W(G))$ then C is also feasible w.r.t. $\text{TGP}(G, T)$ and

$$\text{OPT}(G, W(G)) = \text{OPT}(G, T). \quad (4)$$

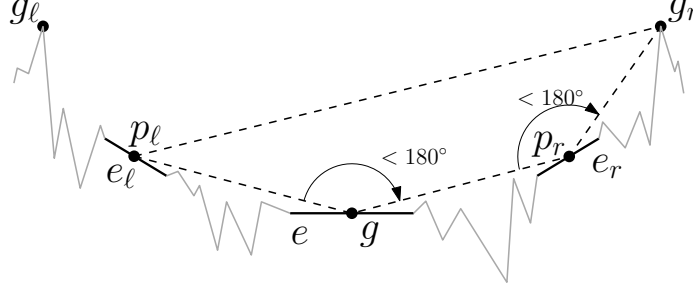
Proof. Let $C \subseteq G$ cover $W(G)$, and consider some point $w \in T$. We show that $w \in \mathcal{V}(C)$. By assumption, $w \in \mathcal{V}(G)$ and thus $w \in f$ for some feature $f \in F_G$. The set $W(G)$ contains some witness in $w_f \in f$, or a witness $w_{f'} \in f'$ with $G(f') \subseteq G(f)$ by construction. In the first case, w must be covered, otherwise w_f would not be covered and C would be infeasible for $\text{TGP}(G, W(G))$. In the second case $w_{f'}$ is covered, so some guard in $G(f')$ is part of C , and that guard also covers f and therefore w .

As for Equation (4), observe that $\text{TGP}(G, W(G))$ is a relaxation of $\text{TGP}(G, T)$, so $\text{OPT}(G, W(G)) \leq \text{OPT}(G, T)$ follows. Furthermore, if C is feasible and optimal w.r.t. $\text{TGP}(G, W(G))$, it is also feasible for $\text{TGP}(G, T)$ as argued above. It follows that $|C| = \text{OPT}(G, W(G)) \geq \text{OPT}(G, T)$, proving (4). \square

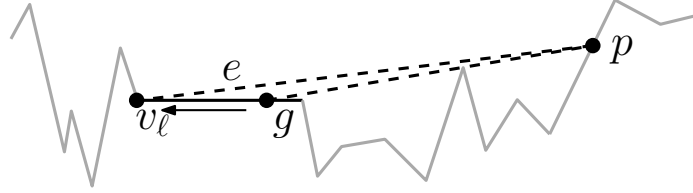
Observation 2.4. *Using the set of one witness per inclusion-minimal feature as in Equation (3) may not reduce the worst-case complexity of $|W(G)| \in O(n|G|)$ witnesses.*



(a) The edge e_i is critical w.r.t. g_ℓ and g_r : The right (left) part of e_i , indicated in blue (red), is seen by g_ℓ (g_r) only.



(b) No guard g is both left- and right-guard. Any point on the critical edge e_ℓ seen by g is also seen by g_r , hence e_ℓ cannot be critical w.r.t. g .



(c) Moving the left-guard g to the left. Any point p that g sees to its right remains visible while moving g towards v_ℓ .

Figure 4: Guard discretization: critical edges (a) and dominated guards (b)–(c).

Proof. See Figure 3(b). For $|G| \in \Theta(n)$ consider the terrain with $\Theta(n)$ valleys with $\frac{|G|}{3}$ guards placed on the left (blue) and the right (red) slope each. In addition there is one guard (black) placed in each valley. Thus, each of the $\Theta(n)$ valleys contains $\Theta(|G|)$ inclusion-minimal intervals depicted in violet, resulting in $O(n|G|)$ inclusion-minimal features. \square

Nevertheless, using only inclusion-minimal witnesses significantly speeds up our implementation, refer to Sections 5.3 and 7.4.6.

Observation 2.5. $W(G)$ does not require any end point p between two maximal intervals I_1 and I_2 : $G(p) = G(I_1) \cup G(I_2)$, since visibility regions are closed sets.

2.2 Guards

Throughout this section, let T be a terrain, $V = V(T)$ its vertices, and $E = E(T)$ its edges. Let $C \subset T$ be feasible w.r.t. $\text{TGP}(T, T)$, i.e., some finite, possibly optimal, guard cover

of T . Define U as all vertices along with the extremal points of their visibility regions:

$$U := V \cup \bigcup_{v \in V} \{p \mid p \text{ is extremal in } \mathcal{V}(v)\}. \quad (5)$$

Observation 2.6. $|U| \in O(n^2)$ as noted by Ben-Moshe et al. [1]: n vertices with visibility regions of $O(n)$ subterrains each.

Ben-Moshe et al. use a similar set, but they also add an arbitrary point of T between each pair of consecutive points in U . They need these points as witnesses. We, however, keep the witnesses separate by our definition of $\text{TGP}(G, W)$.

In the remainder of this section we show that U contains all guard candidates necessary for solving the Continuous Terrain Guarding Problem (CTGP), $\text{TGP}(T, T)$, i.e., that $\text{OPT}(U, T) = \text{OPT}(T, T)$. Our strategy is to show that in any cover C of T it is always possible to move a guard in $C \setminus U$ to a carefully chosen point in U without losing coverage. This procedure preserves the cardinality and feasibility of any feasible cover; iterating it results in a cover $C \subseteq U$. In particular, this is possible for an optimal guard cover.

First observe that an edge that is entirely covered by a guard $g \in C \setminus U$ is still covered after moving g to one of its neighbors in U .

Lemma 2.7. *Let $g \in C \setminus U$ be a guard that covers an entire edge $e_i \in E$. Then u_ℓ and u_r , the U -neighbors of g , with*

$$u_\ell = \max\{u \in U \mid u < g\} \quad (6)$$

$$u_r = \min\{u \in U \mid g < u\} \quad (7)$$

each entirely cover e_i , too.

Proof. g covers e_i , so $v_i, v_{i+1} \in \mathcal{V}(g)$, implying $g \in \mathcal{V}(v_i) \cap \mathcal{V}(v_{i+1})$. Moving g towards u_ℓ does not move g out of $\mathcal{V}(v_i)$ or $\mathcal{V}(v_{i+1})$, as the boundaries of those regions are contained in U by construction. Hence, $v_i, v_{i+1} \in \mathcal{V}(u_\ell)$ and thus $e_i \subseteq \mathcal{V}(u_\ell)$. Analogously $e_i \subseteq \mathcal{V}(u_r)$. \square

It remains to consider the edges not entirely covered by a single guard, refer to Figure 4(a). We refer to such edges as *critical edges*:

Definition 2.8 (Critical Edge). *An edge $e \in E$ is critical w.r.t. $g \in C$ if $C \setminus \{g\}$ covers some part of, but not all of, $\text{int}(e)$. If e is critical w.r.t. some $g \in C$ we call e critical edge.*

So e is critical if and only if more than one guard is responsible for covering $\text{int}(e)$.

Definition 2.9 (Left-Guard/Right-Guard). *$g \in C$ is a left-guard (right-guard) of $e_i \in E$ if $g < v_i$ ($v_{i+1} < g$) and e_i is critical w.r.t. g . We call g a left-guard (right-guard) if it is a left-guard (right-guard) of some $e \in E$.*

For the sake of completeness, we state and prove the following lemma which also follows from the well-established order claim [1]:

Lemma 2.10. *Let $g \in C$ be a guard left of v_i (right of v_{i+1}) such that g covers a non-empty subset of $\text{int}(e_i)$. Then g covers a single interval of e_i , including v_{i+1} (v_i). In particular, this holds if g is a left-guard (right-guard) of e_i .*

Proof. Refer to Figure 4(a). Obviously, $g = g_\ell$ is nowhere below the line supporting e_i . Let p be a point on e_i seen by g_ℓ . It follows that $\overline{g_\ell p}$ and $\overline{p v_{i+1}}$ form an x -monotone convex chain that is nowhere below T . Thus, $\overline{g_\ell v_{i+1}}$ is nowhere below T . It follows that g_ℓ sees v_{i+1} and any point on $\overline{p v_{i+1}}$. A symmetric argument holds for the right-guard g_r . \square

Corollary 2.11. *For a critical edge there is exactly one left- and exactly one right-guard.*

Proof. Suppose for the sake of contradiction that $g, g' \in C$ both are critical left-guards of $e \in E$. By Lemma 2.10, $I := \mathcal{V}(g) \cap e$ and $I' := \mathcal{V}(g') \cap e$ are single intervals on e . Assume w.l.o.g. that $I' \subseteq I$. This contradicts g' being a left-guard of e because g dominates g' on e , i.e., $e \subseteq \mathcal{V}(C \setminus \{g'\})$. So e has exactly one critical left-guard. A symmetric argument shows that e has exactly one right-guard. \square

Corollary 2.12. *Let $e \in E$ be a critical edge and $g_\ell, g_r \in C$ be its left- and right-guards. Then $\mathcal{V}(g_\ell) \cap e \cap \mathcal{V}(g_r) \neq \emptyset$.*

Proof. For the sake of contradiction, suppose $I := e \setminus (\mathcal{V}(g_\ell) \cup \mathcal{V}(g_r)) \neq \emptyset$, and refer to Figure 4(a). Since C is feasible I is covered, so some $g \in C$ sees a point $p \in I$. By Lemma 2.10, g sees a continuous interval containing p and, w.l.o.g., the right vertex of e . It follows that g dominates g_ℓ on e , contradicting that g_ℓ is a critical left-guard of e . \square

By Lemma 2.7, we can move non-critical guards to one of their neighbors in U because they are only responsible for entire edges. Unfortunately, this is impossible if $g \in C \setminus U$ is a left- or a right-guard: We might lose coverage of some part of an edge that is critical w.r.t. g . However, the following lemma establishes that we can move g to its left neighbor vertex if g is not a right-guard (a symmetric version for non-left-guards follows).

Lemma 2.13. *Let C be some finite cover of T , let $g \in C \setminus V$ be a left- but not a right-guard, and let $v_\ell = \max\{v \in V \mid v < g\}$ be the rightmost vertex left of g . Then*

$$C' = (C \setminus \{g\}) \cup \{v_\ell\} \quad (8)$$

is a guard cover of T .

Proof. Since g is a left-guard of some critical edge e_r , there must exist a corresponding right-guard g_r of e_r , see Figure 4(b). Let $p_\ell \in \{p \in \mathcal{V}(g) \mid p \leq g\}$ be a point that g sees to its left. We show that p_ℓ is seen by g_r : Consider p_r , a point in $\mathcal{V}(g) \cap e_r \cap \mathcal{V}(g_r)$, which exists by Corollary 2.12. $\overline{p_\ell g}$, $\overline{g p_r}$, and $\overline{p_r g_r}$ form a convex chain (convex due to $g, p_r \notin V$) that is nowhere below T , so $p_\ell \in \mathcal{V}(g_r)$. Thus, g is dominated to its left by g_r . Moreover, g is dominated to its right by v_ℓ , see Figure 4(c): Let $p \in \{p \in \mathcal{V}(g) \mid g \leq p\}$ be a point seen by g located to its right. Then $\overline{v_\ell g}$ and $\overline{g p}$ form a convex chain nowhere below T , so $p \in \mathcal{V}(v_\ell)$. In conclusion, replacing g by v_ℓ in C yields a feasible cover because $\{p \in \mathcal{V}(g) \mid p \leq g\}$ is covered by g_r and $\{p \in \mathcal{V}(g) \mid g \leq p\}$ by v_ℓ . \square

Corollary 2.14. *Let C be some finite cover of T , let $g \in C \setminus V$ be a right- but no left-guard, and let $v_r = \min\{v \in V \mid g < v\}$ be the leftmost vertex right of g . Then*

$$C' = (C \setminus \{g\}) \cup \{v_r\} \quad (9)$$

is a guard cover of T .

So far, the status is that guards in $C \setminus U$ that are neither left- nor right-guard can be moved to a U -neighbor. Left-guards (right-guards) that are no right-guard (left-guard) can be moved to the next vertex to the left (right). The remaining case, i.e., guards that are both left- and right-guards, cannot happen:

Lemma 2.15. *Let C be a finite cover of T . No $g \in C \setminus V$ is both a left- and a right-guard.*

Proof. Refer to Figure 4(b). Suppose for the sake of contradiction that $g \in C \setminus V$ is the left-guard of an edge e_r (to the right of g) and the right-guard of edge e_ℓ (to the left of g). Since e_r is critical, there must be a right-guard g_r of e_r . By Corollary 2.12 there is a point $p_r \in e_r$ seen by g and g_r . As $g \in C \setminus V$, $g \in \text{int}(e)$ for some edge e .

Now consider some point $p_\ell \in \mathcal{V}(g)$ such that $p_\ell < g$. p_ℓ and p_r are not below the line supported by e and the same holds for g and g_r w.r.t. e_r . It follows that segments $\overline{p_\ell g}$, $\overline{g p_r}$, and $\overline{p_r g_r}$ form an x -monotone convex chain that is nowhere below T . Hence, $p_\ell \in \mathcal{V}(g_r)$. Since p_ℓ was arbitrary, any point $p \in \mathcal{V}(g)$ to the left of g is also seen by g_r , a contradiction to g being a right-guard. \square

The next theorem shows that the set U as defined in Equation (5) contains all guard candidates necessary for a minimum-cardinality guard cover of T .

Theorem 2.16. *Let T be a terrain and consider U from Equation (5). Then we have*

$$\text{OPT}(U, T) = \text{OPT}(T, T). \quad (10)$$

Proof. Let C be optimal w.r.t. $\text{TGP}(T, T)$. We show how to replace a single guard $g \in C \setminus U$ by one in U while maintaining feasibility, i.e., $\mathcal{V}(C) = T$. The claim then follows by induction.

Should g be neither left- nor right-guard, it can be replaced by a neighboring point in U by Lemma 2.7. If g is a left-, but not a right-guard (or vice versa), it can be replaced by its left (right) neighbor in $V \subseteq U$ by Lemma 2.13 (Corollary 2.14). Lemma 2.15 asserts that g cannot be a left- and a right-guard at the same time. \square

2.3 Full Discretization

We formulate the key result of this section: The CTGP, i.e., finding a minimum-cardinality guard cover C guarding an entire terrain T , without any restriction on where on T the guards can be placed, is a discrete problem with a discretization $(U, W(U))$ of size $O(n^3)$.

Theorem 2.17. *Let T be a terrain, and consider U and $W(U)$ from Equations (5) and (3). If $C \subseteq U$ is optimal w.r.t. $\text{TGP}(U, W(U))$, C is optimal w.r.t. $\text{TGP}(T, T)$:*

$$\text{OPT}(T, T) = \text{OPT}(U, W(U)). \quad (11)$$

Proof.

$$\text{OPT}(T, T) \stackrel{(10)}{=} \text{OPT}(U, T) \stackrel{(4)}{=} \text{OPT}(U, W(U)). \quad (12)$$

\square

Observation 2.18. *Observations 2.4 and 2.6 yield: The set of guard candidates U and the witness set $W(U)$ have cardinality $O(n^2)$ and $O(n^3)$, respectively.*

Observation 2.19. *Let B be the largest number of bits required to represent a coordinate of V . The number of bits required to represent the coordinates of a guard candidate $g \in U$ is polynomial in B as the coordinates of g are defined by the intersection of two lines each spanned by two vertices in V .*

3 Complexity Results

For a long time, the NP-hardness of the Continuous Terrain Guarding Problem (CTGP) was generally assumed, but not shown until 2010 by King and Krohn (in the conference version of [33]). In this section we establish that the CTGP is also a member of NP, and thus NP-complete. This is surprising, as it is a long-standing open problem for the more general Art Gallery Problem (AGP): For the AGP it is not known whether the coordinates of an optimal guard cover can be represented with a polynomial number of bits.

Theorem 3.1. *The Continuous Terrain Guarding Problem (CTGP) is NP-complete: Given a terrain T with rational vertices $V(T) \subset \mathbb{Q}^2$ and $k \in \mathbb{N}$, it is NP-complete to decide whether there exist $k \in \mathbb{N}$ guards $G = \{g_1, \dots, g_k\} \subset T$ with $\mathcal{V}(G) = T$.*

Proof. The NP-hardness of the CTGP was established in [33]. It remains to show that the CTGP is a member of NP: A non-deterministic Turing machine determines U (possible in polynomial time by Observation 2.19), and guesses k guards C . It then verifies whether $\mathcal{V}(C) = T$ in polynomial time [26]. \square

4 Polynomial Time Approximation Scheme

In this section we combine our discretization from Section 2 with the Polynomial Time Approximation Scheme (PTAS) for discrete TGP(G, W) with finite $G, W \subset T$ by Gibson et al. [23], who established the following theorem:

Theorem 4.1 (Gibson et al. [23]). *Let T be a terrain, and let $G, W \subset T$ be finite sets of guard candidates and witnesses with $W \subseteq \mathcal{V}(G)$. Then there exists a PTAS for TGP(G, W), i.e., for any constant $\epsilon > 0$, there is an algorithm that returns $C \subseteq G$ with $W \subseteq \mathcal{V}(C)$ and*

$$|C| \leq (1 + \epsilon) \text{OPT}(G, W). \quad (13)$$

We combine our discretization from Theorem 2.17 with Theorem 4.1:

Theorem 4.2. *There is a PTAS for the Continuous Terrain Guarding Problem (CTGP). That is, for any constant $\epsilon > 0$, there is a polynomial-time algorithm which, given a terrain T , returns $C \subset T$ with $\mathcal{V}(C) = T$ and $|C| \leq (1 + \epsilon) \text{OPT}(T, T)$.*

Proof. Using Equations (5) and (3) we determine the sets U and $W(U)$ for T with $|U| + |W(U)| \in O(n^3)$ by Observation 2.18. By Theorem 4.1, we can compute $C \subseteq U \subset T$ with

$$|C| \stackrel{(13)}{\leq} (1 + \epsilon) \text{OPT}(U, W(U)) \stackrel{(11)}{=} (1 + \epsilon) \text{OPT}(T, T), \quad (14)$$

where C is feasible w.r.t. TGP(T, T) by Theorem 2.17. \square

5 Reducing the Size of the Discretization

While $O(n^2)$ guard candidates and $O(n^3)$ witnesses, see Observation 2.18, may be satisfactory from a theoretical point of view, it is imperative to reduce their numbers for an efficient implementation. We propose filtering techniques that, while not reducing the asymptotic size of the discretization, typically remove around 90 % of the guard candidates and an even larger fraction of the witnesses. Experiments in Section 7 demonstrate this to be a key success factor that increases the solvable instance size by several orders of magnitude.

We say that $g \in T$ *dominates* $g' \in T$ if $\mathcal{V}(g') \subseteq \mathcal{V}(g)$, in which case g' can be safely discarded; our filters in Sections 5.1 and 5.2 do just that. A core issue, however, is that visibility calculations are expensive, so the key challenge is to identify dominated guard candidates *without determining their visibility region*. The guard filter in Section 5.2 has that feature. Sections 5.3 and 5.4 discuss witness filtering and an open problem.

5.1 Filtering Dominated Guards

Let T be a terrain and $G \subset T$ a finite set of guard candidates with $\mathcal{V}(G) = T$. Consider $g, g' \in G$, suppose we know $\mathcal{V}(g)$ and $\mathcal{V}(g')$, and observe that checking whether g dominates g' takes $O(n)$ time since visibility regions consist of $O(n)$ subterrains. Moreover, removing all dominated guards from G requires $O(|G|^2)$ domination queries, i.e., an intolerable $O(n^5)$ time when applied to $G = U$ from Equation (5).

Instead, we devise a heuristic using $O(|G|)$ domination queries and thus an acceptable $O(n^3)$ time for $G = U$. Suppose G is ordered w.r.t. x -coordinates. The *local domination filter* removes all guard candidates that are dominated by one of their neighbors. This is based on the observation that neighboring guards' visibility regions often are quite similar or one clearly dominates the other (a local “dent”). Experiments demonstrate that this strategy is beneficial in terms of time and memory consumption, see Section 7.4.5.

5.2 Filtering Edge-Interior Guards

Let T be a terrain and U the guard candidates from Equation (5) and fix an edge e . By Lemma 2.13, Corollary 2.14, and Lemma 2.15 assume w.l.o.g. that all critical guards are located at the vertices. Hence, guards in $U_e := U \cap \text{int}(e)$ are only responsible for covering entire edges. Recall that when moving across $u \in U_e$, a vertex becomes visible or invisible, depending on the direction, by construction of U . Furthermore, covering an entire edge is equivalent to seeing both its vertices. The sets of edges entirely seen by each $u \in U_e$,

$$E_u := \{e \in E \mid e \subseteq \mathcal{V}(u)\} = \{e_i \in E \mid v_i, v_{i+1} \in \mathcal{V}(u)\}, \quad (15)$$

define a partial ordering on U_e w.r.t. inclusion as indicated in Figure 5. Most importantly, u is inclusion-maximal if $E_{u'} \not\supseteq E_u$ for all $u' \in U_e$. We show that it suffices to consider the guard candidates that are inclusion-maximal w.r.t. E_u :

Theorem 5.1. *Let $U'_e \subseteq U_e$ be the set that only contains inclusion-maximal guard candidates w.r.t. entire edges, as defined above. Then*

$$U' = (U \setminus U_e) \cup U'_e \quad (16)$$

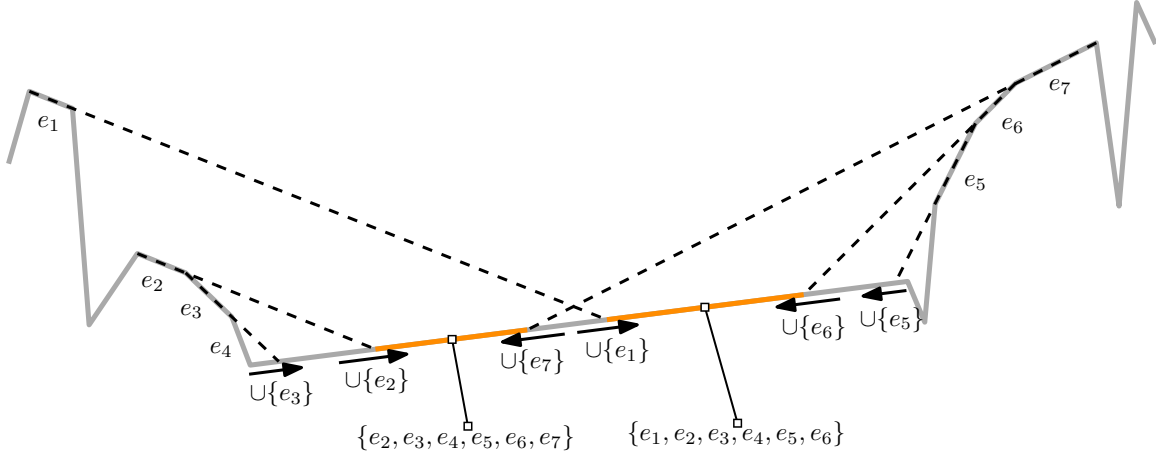


Figure 5: Edge-interior guards are only responsible for entire edges. Edges can only become visible when crossing some $u \in U$, the arrows indicate in which direction. Only the orange regions contain guard candidates that are inclusion-maximal w.r.t. entire edges.

admits covering T with the same number of guards as U , i.e.,

$$\text{OPT}(U', T) = \text{OPT}(U, T). \quad (17)$$

Proof. A guard cannot be left- and right-guard at the same time by Lemma 2.15. Furthermore, by Lemma 2.13 (Corollary 2.14), a left-guard (right-guard) can be moved to its left (right) neighbor in V . Thus, w.l.o.g., $u \in U_e$ is no left- or right-guard, because U_e does not contain vertices by definition. Hence, no edge is critical w.r.t. u by Definition 2.8, so u is only responsible for covering entire edges and can be replaced by its inclusion-maximal sibling in U'_e without changing the feasibility or cardinality of a cover of T . \square

The key is that identifying guard candidates $u \in U \setminus V$ that are not inclusion-maximal w.r.t. entire edges can be implemented without determining $\mathcal{V}(u)$: For each $u \in U \setminus V$, store a reference to which vertex's visibility region is extremal at u , as well as whether it is situated left or right of u . This allows to decide which vertex becomes visible or invisible when sweeping across u from left to right, as indicated in Figure 5.

We use the following sweep line algorithm. For every $e \in E$, sweep through U_e from left to right. While encountering $u \in U_e$ where new vertices become visible, do nothing. When reaching the first $u \in U_e$ where a vertex becomes invisible, report u . Since u is inclusion-maximal w.r.t. vertices, it is inclusion-maximal w.r.t. entire edges. Then ignore all points corresponding to vertices becoming invisible until encountering the first that becomes visible, and continue as above.

This discards up to 98 % of the guard candidates efficiently enough to essentially remove the computational boundary between Terrain Guarding Problem with Vertex Guards (VTGP) and Continuous Terrain Guarding Problem (CTGP), see Section 7.4.4.

Observation 5.2. *The above sweep line algorithm needs only the visibility regions of vertices, not those of $U \setminus V$. Deciding whether a vertex v becomes visible or invisible at $u \in U_e$ depends only on whether u is extremal in $\mathcal{V}(v)$ and on $v < u$, as described above.*

Observation 5.3. *Filtering U as above still yields $O(n^2)$ guard candidates: Insert a vertex below each guard on the slopes in Figure 3(b). Then every other interval is inclusion-maximal w.r.t. the vertices on the slopes.*

5.3 Filtering Witnesses

Let U be a possibly filtered set of guard candidates. The construction of the witness set $W(U)$ as in Equation (3) already includes a filtering mechanism: only inclusion-minimal witnesses need to be kept. Observe that a smaller, filtered, U automatically yields a smaller $W(U)$. Furthermore, observe that in terms of an implementation witnesses are much cheaper than guard candidates: They require no visibility region or coordinates — by Observation 2.2, they only need to store references to the guards covering them.

We acquire witnesses very much like in Section 5.2: Sort the extremal points of all guard candidates' visibility regions by their x -coordinates. For each of these points we know whether a visibility region opens or closes and to which guard it is associated. Sweeping through these points, it is straightforward to keep track of which guard candidates see the current event point and where this set is inclusion-minimal.

Our approach keeps witnesses that are locally, but not necessarily globally, inclusion-minimal. We can efficiently exploit the underlying geometry to identify the locally inclusion-minimal witnesses, but it is an open question whether globally inclusion-minimal witnesses can be identified just as efficiently. However, experiments demonstrate that our approach is extremely effective, see Section 7.4.6.

5.4 Open Problem

We would like to find an optimal discretization. But what is an optimal discretization? Obviously, a good discretization is small — asymptotically and, if an implementation is of interest, in terms of constant factors. However, a discretization that minimizes $|G| + |W|$ is one where $|G| = \text{OPT}(T, T)$, i.e., just as hard to find as solving $\text{TGP}(T, T)$. Hence, we require a discretization to be obtainable in polynomial time.

Our discretization has size $|U| + |W(U)| \in O(n^3)$. The filters do not reduce the asymptotic complexity but prove effective by reducing, on average, the size by more than 90%. Is there a discretization, obtainable in polynomial time, of size $o(n^3)$?

6 Optimal Solutions with Integer Linear Programming

We combine discretization and filters from Sections 2 and 5 to an efficient algorithm. It solves instances of the Terrain Guarding Problem (TGP) with up to 10^6 vertices within roughly 1–2 minutes on a standard desktop computer, see Section 7. For evaluation, the filtering techniques can be enabled individually.

input : Terrain T	
output : Guard cover of T	
1: $(U, W) \leftarrow (V(T), \emptyset)$	\triangleright vertices are guard candidates in both modes
2: for $u \in U$ do	
3: \lfloor determine $\mathcal{V}(u)$	
4: if POINTGUARDS then	\triangleright as opposed to VERTEXGUARDS
5: $U \leftarrow U \cup \bigcup_{v \in V(T)} \{p \mid p \text{ is extremal in } \mathcal{V}(v)\}$	\triangleright Equation (5)
6: if EDGEFILTER then	
7: \lfloor filter edge-interior guards in U by sweep	\triangleright Section 5.2
8: for $u \in U \setminus V(T)$ do	
9: \lfloor determine $\mathcal{V}(u)$	\triangleright after EDGEFILTER, see Section 5.2
10: if DOMINATIONFILTER then	
11: \lfloor filter out guards in U dominated by a neighbor	\triangleright Section 5.1
12: if WITNESSFILTER then	
13: $W \leftarrow$ inclusion-minimal features from overlay of U	\triangleright Equation (3)
14: else	
15: $W \leftarrow$ all features from overlay of U	\triangleright unfiltered version of Equation (3)
16: solve TGP(U, W) with an IP solver	

Algorithm 1: Optimal solutions for the TGP.

6.1 IP Formulation

Let T be a terrain, and let $G, W \subset T$ be finite sets of guard candidates and witnesses, such that $W \subseteq \mathcal{V}(G)$. We formulate TGP(G, W) as Integer Linear Program (IP):

$$\min \sum_{g \in G} x_g \quad (18)$$

$$\text{s.t.} \quad \sum_{g \in \mathcal{V}(w) \cap G} x_g \geq 1 \quad \forall w \in W \quad (19)$$

$$x_g \in \{0, 1\} \quad \forall g \in G. \quad (20)$$

A binary variable x_g for each guard candidate $g \in G$ indicates whether g is picked: $x_g = 1$ if and only if g is part of the cover. For each witness $w \in W$, a constraint ensures that w is covered by at least one guard. We minimize the number of guards in the cover.

Choosing $G = U$ (possibly filtered) and $W = W(U)$ from Equations (5) and (3), (18)–(20) model the Continuous Terrain Guarding Problem (CTGP); picking $G = V$ and $W = W(V)$ corresponds to the Terrain Guarding Problem with Vertex Guards (VTGP).

6.2 Algorithm

Algorithm 1 has two modes: POINTGUARDS for solving TGP(T, T), and VERTEXGUARDS for TGP(V, T). Everything except lines 4–9 applies to both modes, lines 4–9 generate non-vertex guard candidates and possibly filter them.

Filtering mechanisms are activated individually: `DOMINATIONFILTER` from Section 5.1 removes guard candidates that are dominated by one of their neighbors, `EDGEFILTER` corresponds to the guard filter from Section 5.2 and is only available in the `POINTGUARDS` mode, and `WITNESSFILTER` determines whether only the inclusion-minimal witness are used, refer to Equation (3).

We remark two things about line 16. (1) It is the only subroutine that requires exponential time, due to the NP-hardness of the TGP [33]. In our experiments, however, this is not the bottleneck of our algorithm; the geometric subroutines require most time and memory. We discuss this in Sections 7.4.7 and 7.4.8. (2) Algorithm 1 transforms an instance of VTGP or CTGP into an instance of Set Cover (SC) which it entrusts to a solver. An IP or SAT solver, a SC approximation algorithm, the Polynomial Time Approximation Scheme (PTAS) by Gibson et al. [23], or any other solver (including those oblivious to the underlying geometry) would work. Observe that all solvers benefit from our filtering framework. However, since benchmarking the underlying solver is not our concern, we restrict our experiments to a state-of-the-art IP solver.

6.3 Implementation

We implemented Algorithm 1 in C++11 and compiled with g++-4.8.4 [22]. The geometric subroutines use CGAL-4.6 [4] (Computational Geometry Algorithms Library) with the `CGAL::Exact_predicates_exact_constructions_kernel` kernel; note that we follow the Exact Geometric Computation (EGC) paradigm, i.e., use exact number types instead of floating-point arithmetic in geometric subroutines, to guarantee correctness. Terrain visibility is solved by the implementation of Haas and Hemmer [25]. We solve IPs using CPLEX-12.6.0 [9]. Furthermore, we use boost-1.58.0 [2] and simple-svg-1.0.0 [43].

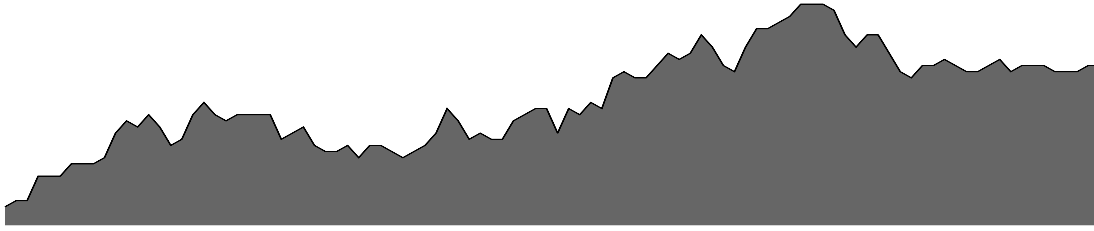
7 Experiments

We evaluate Algorithm 1. It can solve large instances within minutes on a standard desktop computer; our filtering techniques prove critical to success. Especially `EDGEFILTER` and `WITNESSFILTER`, see Sections 5.2 and 5.3, significantly increase the solvable instance size. Our instances, the tested configurations of Algorithm 1, the experimental setup, and our findings are described in Sections 7.1, 7.2, 7.3, and 7.4, respectively.

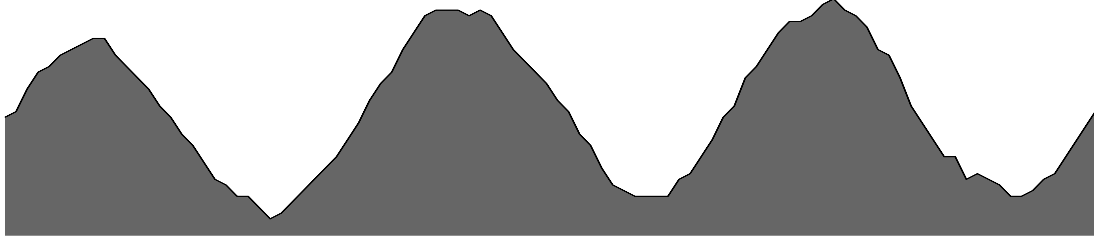
7.1 Instances

We test four classes of random terrains from the 2015-08-06 version of the Terrain Guarding Problem Instance Library (TGPIIL) [21], see Figure 6 for an overview. Each class comprises 20 instances with 10^3 , 10^4 , 10^5 , $5 \cdot 10^5$, and 10^6 vertices each, yielding 400 instances.

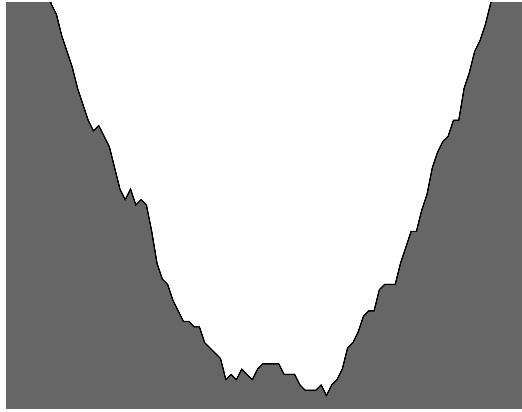
A WALK, see Figure 6(a), has n vertices with x -coordinates $0, \dots, n-1$, and the $(i+1)$ -th y -coordinate is a random offset from the i -th. SINEWALK and PARABOLAWALK, see Figures 6(b) and 6(c), are the sum of a WALK and a properly scaled sine or parabola, respectively. Both classes pose a challenge because many points see a large slope, highly fragmented by shadows of local features.



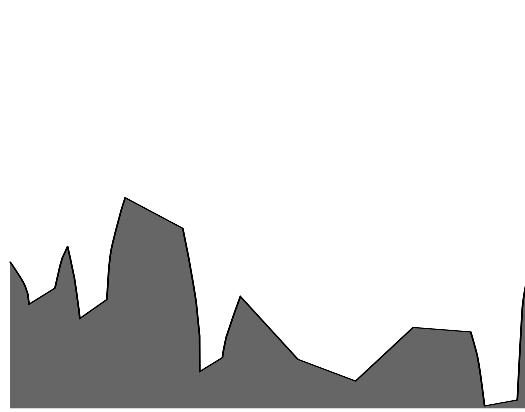
(a) WALK: Random walk with uniform step width.



(b) SINEWALK: Sum of a sine wave and a random walk.



(c) PARABOLAWALK: Sum of a parabola and a random walk.



(d) CONCAVEVALLEYS: Optimal solutions require point guards in the valley centers.

Figure 6: Four classes of randomly generated test instances available in the TGPIIL [21].

Preliminary experiments revealed that the above classes hardly require non-vertex guards for optimal solutions. Hence we propose the CONCAVEVALLEYS class, see Figure 6(d), which encourages point guards. An instance starts as a WALK. Iteratively pick a random edge and replace it by a valley with concave slopes. Connect the slopes by a bottom edge, such that a point in its interior covers both slopes as in Figure 1(b). An optimal solution for such a terrain usually requires guards in bottom edges' interiors.

None of the above classes deliberately provokes the NP-hardness of the Terrain Guarding Problem (TGP); they are not designed to contain a reduction of hard instances of e.g. PLANAR 3SAT, as used in the NP-hardness proof of King and Krohn [33]. In our case, testing such instances is out of scope: We provide and evaluate the means to transform a terrain into a small discretization that can be handed to a solver; Integer Linear Program (IP), Polynomial Time Approximation Scheme (PTAS), SAT, or other. The transformation

Configuration	Mode	EDGEFILTER	DOMINATIONFILTER	WITNESSFILTER
VDEFAULT	VERTEXGUARDS	n/a	yes	yes
VNoDOM	VERTEXGUARDS	n/a	no	yes
VNoW	VERTEXGUARDS	n/a	yes	no
PDEFAULT	POINTGUARDS	yes	yes	yes
PNoEDGE	POINTGUARDS	no	yes	yes
PNoDOM	POINTGUARDS	yes	no	yes
PNoW	POINTGUARDS	yes	yes	no

Table 1: Algorithm configurations, VTGP above and CTGP below.

has to be efficient and our experiments are designed to verify just that. Combinatorially hard instances merely benchmark the underlying solver.

7.2 Configurations

We test Algorithm 1 in seven configurations, see Table 1, to individually assess the impact of each filtering technique from Section 5. VDEFAULT, VNoDOM, and VNoW test the VERTEXGUARDS mode; PDEFAULT, PNoEDGE, PNoDOM, and PNoW test the considerably harder POINTGUARDS mode. Recall that EDGEFILTER does not apply to VERTEXGUARDS mode. Since we test 400 instances, this results in 2800 test runs.

7.3 Experimental Setup

We used eight identical Linux 3.13 machines with Intel Core i7-3770 CPUs running at 3.4 GHz, provided with 8 MB of cache and 16 GB of main memory. Every run was limited to 15 minutes of CPU time and 14 GB of memory. Our software, except solving IPs with CPLEX, is not parallelized. Refer to Section 6.3 for details regarding the toolchain.

7.4 Results

The solution rates and median solution times of every combination of configuration, instance class, and instance complexity are listed in Tables 2 and 3; each cell corresponds to 20 test runs. Due to the imposed time and memory limits, not all test runs succeeded; we account for unfinished test runs with an infinite completion time. Hence, we use the median instead of the mean throughout the analysis. More fine-grained timing information is presented in Figure 9. Except in PNoEDGE mode, all unsolved instances were caused by running out of memory, see Section 7.4.8. We dedicate one subsection each to the relative hardness of the instance classes (Section 7.4.1), an overview of VERTEXGUARDS and POINTGUARDS modes (Sections 7.4.2 and 7.4.3), the impact of EDGEFILTER, DOMINATIONFILTER and WITNESSFILTER (Sections 7.4.4, 7.4.5 and 7.4.6), timing behavior (Section 7.4.7), and memory consumption (Section 7.4.8).

7.4.1 Overview: Instances

Tables 2 and 3 clearly reveal that SINEWALK and PARABOLAWALK are harder to solve than WALK and CONCAVEVALLEYS. This is to be expected, since SINEWALK and PARABOLA-

Configuration	Instance	#vertices				
		10^3	10^4	10^5	$5 \cdot 10^5$	10^6
VDEFAULT	WALK	100 %	100 %	100 %	100 %	100 %
	SINEWALK	100 %	100 %	100 %	0 %	0 %
	PARABOLA WALK	100 %	100 %	100 %	0 %	0 %
	CONCAVE VALLEYS	100 %	100 %	100 %	100 %	100 %
VNoDOM	WALK	100 %	100 %	100 %	100 %	100 %
	SINEWALK	100 %	100 %	100 %	0 %	0 %
	PARABOLA WALK	100 %	100 %	100 %	0 %	0 %
	CONCAVE VALLEYS	100 %	100 %	100 %	100 %	100 %
VNoW	WALK	100 %	100 %	100 %	0 %	0 %
	SINEWALK	100 %	100 %	0 %	0 %	0 %
	PARABOLA WALK	100 %	25 %	0 %	0 %	0 %
	CONCAVE VALLEYS	100 %	100 %	100 %	0 %	0 %
PDEFAULT	WALK	100 %	100 %	100 %	100 %	100 %
	SINEWALK	100 %	100 %	100 %	0 %	0 %
	PARABOLA WALK	100 %	100 %	100 %	0 %	0 %
	CONCAVE VALLEYS	100 %	100 %	100 %	100 %	100 %
PNoEDGE	WALK	100 %	100 %	100 %	55 %	0 %
	SINEWALK	100 %	100 %	0 %	0 %	0 %
	PARABOLA WALK	100 %	100 %	0 %	0 %	0 %
	CONCAVE VALLEYS	100 %	100 %	100 %	90 %	0 %
PNoDOM	WALK	100 %	100 %	100 %	100 %	100 %
	SINEWALK	100 %	100 %	100 %	0 %	0 %
	PARABOLA WALK	100 %	100 %	100 %	0 %	0 %
	CONCAVE VALLEYS	100 %	100 %	100 %	100 %	100 %
PNoW	WALK	100 %	100 %	100 %	0 %	0 %
	SINEWALK	100 %	100 %	0 %	0 %	0 %
	PARABOLA WALK	100 %	20 %	0 %	0 %	0 %
	CONCAVE VALLEYS	100 %	100 %	100 %	0 %	0 %

Table 2: Solution rates for each configuration, instance class, and instance complexity.

WALK contain a WALK as additive noise, and since the sole purpose of CONCAVE VALLEYS is to encourage placing non-vertex guards, see Section 7.1. Furthermore, SINEWALK and PARABOLA WALK comprise facing valleys, resulting in highly fragmented visibility regions and complex visibility overlays in which a large portion of the guards and witnesses cannot be filtered out. This induces time and memory intensive calculations and a complex IP, making SINEWALK and PARABOLA WALK challenging instance classes.

7.4.2 Overview: Vertex Guards

VDEFAULT and VNoDOM solve all instances of WALK and CONCAVE VALLEYS, and the SINEWALK and PARABOLA WALK instances of up to 10^5 vertices; VNoW can solve instances which are smaller by about a factor of 10, see Table 2. This already demonstrates the importance of WITNESSFILTER. VDEFAULT and VNoDOM have comparable running times, with a slight advantage for VDEFAULT, see Table 3; VNoW is slower.

Configuration	Instance	#vertices				
		10 ³	10 ⁴	10 ⁵	5 · 10 ⁵	10 ⁶
VDEFAULT	WALK	0.0 s	0.2 s	2.9 s	18.0 s	40.3 s
	SINEWALK	0.0 s	0.9 s	13.8 s	n/a	n/a
	PARABOLA WALK	0.1 s	1.8 s	21.7 s	n/a	n/a
	CONCAVE VALLEYS	0.2 s	2.4 s	24.2 s	177.2 s	337.6 s
VNoDOM	WALK	0.0 s	0.3 s	3.6 s	21.8 s	48.6 s
	SINEWALK	0.1 s	1.3 s	18.2 s	n/a	n/a
	PARABOLA WALK	0.1 s	2.5 s	27.8 s	n/a	n/a
	CONCAVE VALLEYS	0.2 s	2.4 s	24.3 s	177.1 s	411.0 s
VNoW	WALK	0.0 s	0.4 s	8.8 s	n/a	n/a
	SINEWALK	0.1 s	6.4 s	n/a	n/a	n/a
	PARABOLA WALK	0.4 s	n/a	n/a	n/a	n/a
	CONCAVE VALLEYS	0.2 s	2.7 s	32.0 s	n/a	n/a
PDEFAULT	WALK	0.0 s	0.3 s	4.6 s	27.9 s	62.4 s
	SINEWALK	0.1 s	1.7 s	26.3 s	n/a	n/a
	PARABOLA WALK	0.2 s	3.3 s	45.1 s	n/a	n/a
	CONCAVE VALLEYS	0.1 s	0.8 s	10.3 s	68.2 s	137.4 s
PNoEDGE	WALK	0.2 s	4.5 s	79.7 s	833.3 s	n/a
	SINEWALK	1.0 s	58.8 s	n/a	n/a	n/a
	PARABOLA WALK	4.1 s	220.3 s	n/a	n/a	n/a
	CONCAVE VALLEYS	0.2 s	3.2 s	58.3 s	652.3 s	n/a
PNoDOM	WALK	0.0 s	0.4 s	5.0 s	31.5 s	70.4 s
	SINEWALK	0.1 s	2.0 s	31.1 s	n/a	n/a
	PARABOLA WALK	0.2 s	4.0 s	51.8 s	n/a	n/a
	CONCAVE VALLEYS	0.1 s	0.9 s	10.7 s	68.0 s	145.5 s
PNoW	WALK	0.0 s	0.6 s	10.4 s	n/a	n/a
	SINEWALK	0.1 s	7.8 s	n/a	n/a	n/a
	PARABOLA WALK	0.6 s	n/a	n/a	n/a	n/a
	CONCAVE VALLEYS	0.1 s	1.1 s	20.8 s	n/a	n/a

Table 3: Median solution times per configuration, instance class, and instance complexity.

7.4.3 Overview: Point Guards

In terms of solved instances, refer to Table 2, PDEFAULT and PNoDOM are the strongest configurations, solving all WALK and CONCAVE VALLEYS instances as well as the SINEWALK and PARABOLA WALK instances with up to 10⁵ vertices. PNoW and PNoEDGE are much weaker. Table 3 indicates that PDEFAULT is slightly faster than PNoDOM. It is clear from the performance of PNoEDGE that EDGEFILTER is crucial in POINTGUARDS mode.

7.4.4 Impact of Filtering Edge-Interior Guards

Recall that EDGEFILTER, see Section 5.2, only applies to POINTGUARDS mode. Table 4 depicts the percentage of guards it removes in the PNoDOM configuration—the only configuration without interfering guard filters—and Figure 7 illustrates its effectiveness using a 10⁵-vertex PARABOLA WALK as example.

EDGEFILTER proves to be our most effective guard filter by removing roughly 90 % (80 %) of the guard candidates in the 10⁶ vertex WALK (CONCAVE VALLEYS) instances, and

Configuration	Instance	#vertices				
		10^3	10^4	10^5	$5 \cdot 10^5$	10^6
PNoDom	WALK	80.1 %	86.8 %	89.5 %	91.0 %	91.7 %
	SINEWALK	92.7 %	97.6 %	98.3 %	n/a	n/a
	PARABOLAWALK	97.5 %	98.8 %	98.9 %	n/a	n/a
	CONCAVEVALLEYS	65.8 %	72.5 %	77.7 %	79.9 %	80.6 %

Table 4: Median percentage of guard candidates removed by EDGEFILTER.

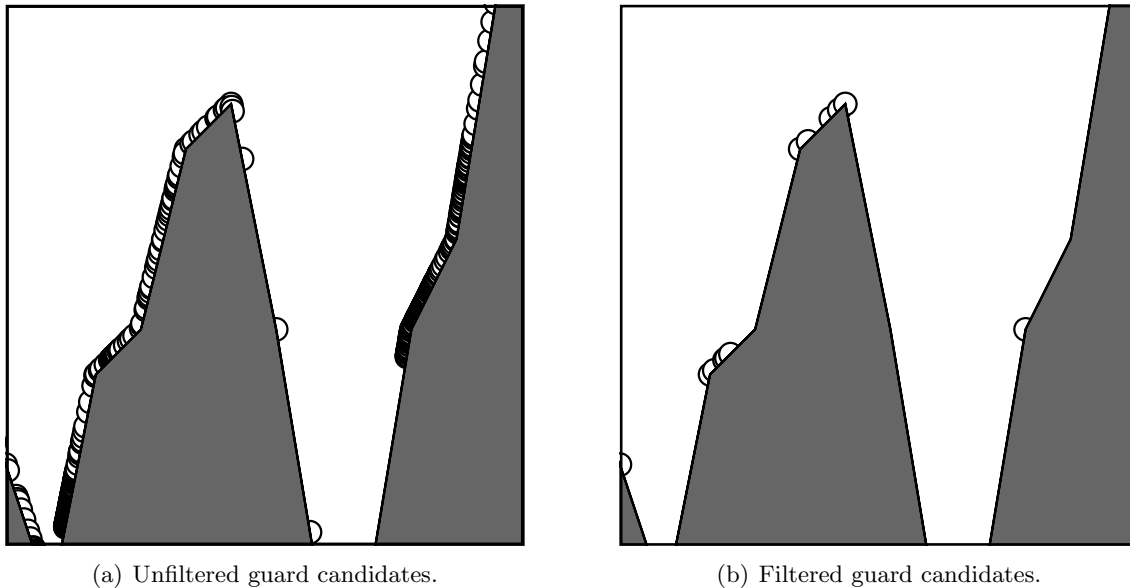


Figure 7: The effect of EDGEFILTER (excerpt of a 10^5 -vertex PARABOLAWALK). White circles represent guard candidates.

well above 95 % in the largest solved SINEWALK and PARABOLAWALK instances. Tables 2 and 3 demonstrate that EDGEFILTER massively improves performance in terms of solution rates and median solution times. This makes it the key success factor when solving the Continuous Terrain Guarding Problem (CTGP), removing the computational barrier between Terrain Guarding Problem with Vertex Guards (VTGP) and CTGP: Without it, PNoEDGE would be the state of the art, and solvable instances of the CTGP would be smaller by at least a factor of 10 than for VTGP and would take much more time.

7.4.5 Impact of Filtering Dominated Guards

Table 5 displays the percentage of guard candidates that DOMINATIONFILTER, refer to Section 5.1, filtered out in the VDEFAULT and PNoEDGE configurations. Observe that we need to obtain these numbers in configurations without other active filters.

DOMINATIONFILTER has no impact on the solution rates within the limits imposed by our setup (see Section 7.3). VDEFAULT and PDEFAULT are slightly faster than VNoDOM and PNoDOM, respectively. However, the key advantage of DOMINATIONFILTER is that it saves memory by deleting dominated guard candidates; this is important since memory

Configuration	Instance	#vertices				
		10^3	10^4	10^5	$5 \cdot 10^5$	10^6
VDEFAULT	WALK	65.8 %	64.1 %	63.6 %	63.5 %	63.5 %
	SINEWALK	58.5 %	63.0 %	63.6 %	n/a	n/a
	PARABOLA WALK	59.9 %	63.3 %	63.6 %	n/a	n/a
	CONCAVE VALLEYS	13.4 %	13.1 %	13.3 %	13.3 %	13.3 %
PNoEDGE	WALK	92.9 %	94.7 %	95.6 %	95.8 %	n/a
	SINEWALK	88.1 %	96.7 %	n/a	n/a	n/a
	PARABOLA WALK	93.2 %	98.0 %	n/a	n/a	n/a
	CONCAVE VALLEYS	77.1 %	80.5 %	83.9 %	85.0 %	n/a

Table 5: Median percentage of guard candidates removed by DOMINATIONFILTER.

Configuration	Instance	#vertices				
		10^3	10^4	10^5	$5 \cdot 10^5$	10^6
VDEFAULT	WALK	91.9 %	94.3 %	95.4 %	96.1 %	96.4 %
	SINEWALK	95.8 %	98.8 %	99.3 %	n/a	n/a
	PARABOLA WALK	98.6 %	99.4 %	99.5 %	n/a	n/a
	CONCAVE VALLEYS	76.5 %	80.5 %	83.7 %	85.1 %	85.5 %
PDEFAULT	WALK	91.9 %	94.3 %	95.5 %	96.1 %	96.4 %
	SINEWALK	96.3 %	98.9 %	99.3 %	n/a	n/a
	PARABOLA WALK	98.7 %	99.5 %	99.5 %	n/a	n/a
	CONCAVE VALLEYS	80.3 %	84.2 %	87.4 %	88.7 %	89.1 %

Table 6: Median percentage of witnesses removed by WITNESSFILTER.

consumption is the bottleneck of our implementation, see Section 7.4.8.

7.4.6 Impact of Filtering Witnesses

The percentage of witnesses removed by WITNESSFILTER, see Section 5.3, in the VDEFAULT and PDEFAULT configurations is displayed in Table 6. Throughout our instances, WITNESSFILTER removes the vast majority of witnesses, often more than 95 %. Furthermore, Table 2 clearly shows that disabling it reduces the solvable instance size by at least a factor of 10. All of this renders WITNESSFILTER simple, fast, and useful.

7.4.7 Timing Behavior

Figures 8 and 9 show how much CPU time is spent in each part of Algorithm 1 and the distribution of the solution times, respectively. For a comparison, we pick an instance class and a complexity that was solved by every configuration: WALK with 10^5 vertices—the other combinations are left out, as they permit the same interpretation.

The strongest impact is that of EDGEFILTER, which is disabled in the rightmost bar in Figure 8. Without EDGEFILTER there is a computational gap between POINTGUARDS and VERTEXGUARDS mode, as determining visibility regions of unneeded guards dominates the CPU time. Guard-filtering time also increases in PNoEDGE mode because DOMINATIONFILTER is active and has more guards to compare.

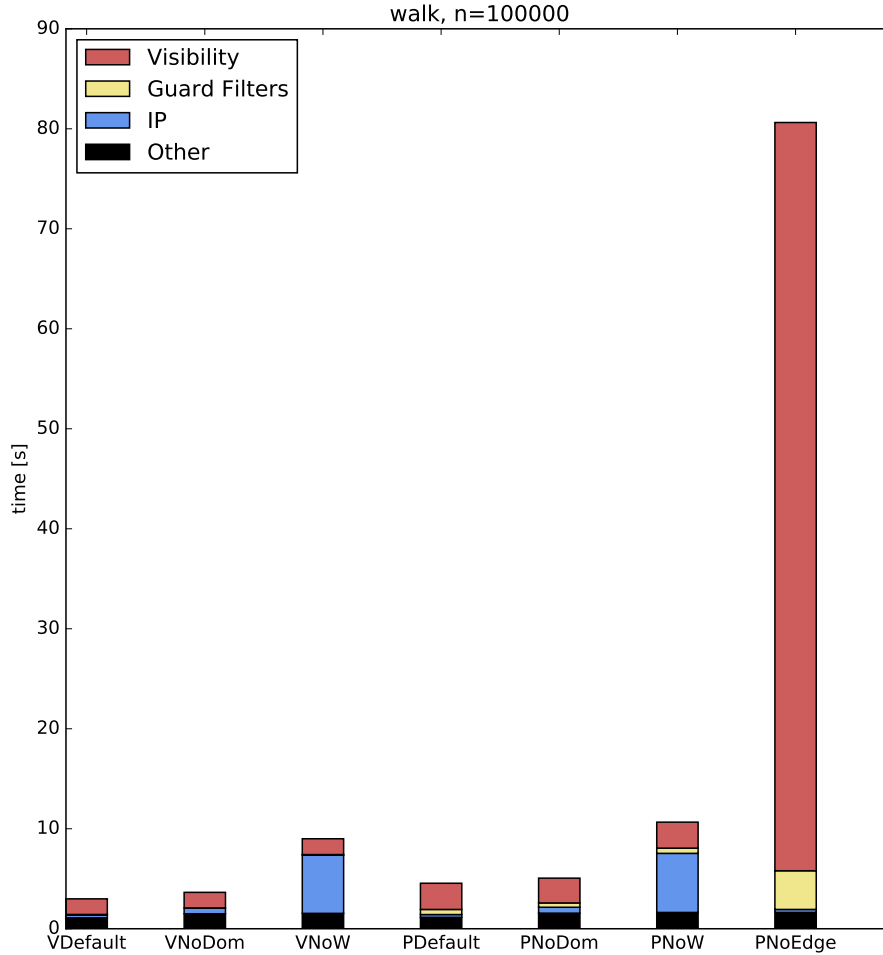


Figure 8: Median CPU time spent by subroutine regarding 10^5 -vertex WALK instances.

WITNESSFILTER has the second-most important impact. In its absence, CPU times roughly double due to increased IP solution times: In the IP, non-inclusion-minimal witnesses form constraints that are dominated by the inclusion-minimal witnesses' constraints. The IP solver eliminates dominated constraints in a preprocessing phase, but WITNESSFILTER does so more efficiently since it exploits the underlying geometry.

The timing behavior is hardly influenced by DOMINATIONFILTER. It is, however, beneficial w.r.t. memory consumption, see Section 7.4.8.

A general observation is that the filtering mechanisms significantly reduce the computational overhead. One would expect IP solution times to dominate an exact solver for an NP-hard problem. This, however, tends not to be the case in geometric optimization problems like the Art Gallery Problem (AGP) and its relatives [10]. It stems from the Exact Geometric Computation (EGC) paradigm: Instead of floating-point arithmetic, exact number types must be used to ensure correct results. VDEFAULT and PDEFAULT still are not clearly dominated by IP solution times, but much closer to it than unfiltered approaches.

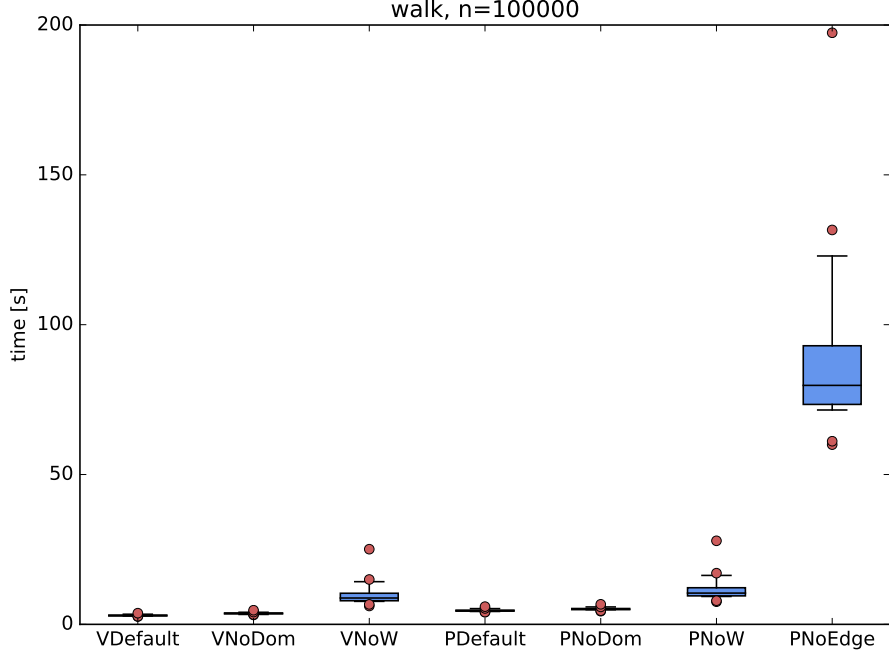


Figure 9: Box plot of solution times regarding 10^5 -vertex WALK instances. Boxes comprise first, second, and third quartile; whiskers indicate the 10th and 90th percentile.

7.4.8 Memory Consumption

Within our experimental setup, using the VDEFAULT and PDEFAULT modes, even instances with 10^6 vertices are solved within minutes. All unsolved configuration/instance pairs run out of memory, not time; only PNOEDGE occasionally runs out of time—owed to the large number of unnecessary visibility calculations otherwise prevented by EDGEFILTER, see Sections 7.4.4 and 7.4.7. So as long as instances are not designed to reveal the NP-hardness of the TGP, the limiting resource is memory.

Two phases of Algorithm 1 generate a significant amount of data which persists in memory. The first phase is the computation of all visibility regions of all vertices V in line 3. This stores $O(n^2)$ x -coordinates in memory which define the unfiltered guard candidate set U . Filters remove the vast majority of candidates from U . The second phase determines the visibility regions of the remaining points in $U \setminus V$, generating the largest chunk of data in line 9: At this point we keep $O(n^3)$ x -coordinates in memory. We conjecture that simultaneously holding all these visibility regions in memory cannot be avoided since a guard at the far right of the terrain may still see a region at its very left; hence we do not see a way to apply WITNESSFILTER before knowing all extremal points. As a lower bound, observe that a discretization with guards G and witnesses W yields a constraint matrix $A \in \{0, 1\}^{|W| \times |G|}$ of the IP (18)–(20) with $A_{wg} = 1 \Leftrightarrow w \in \mathcal{V}(g)$, i.e., one with $O(n^5)$ entries for the CTGP.

We remark that the memory bottleneck is amplified by the fact that we follow the EGC paradigm, which ensures a correct and consistent representation of all visibility re-

gions and a correct order of all visibility events. Specifically, we do not store coordinates of points using floating-point arithmetic. Neither do we use the other extreme, i.e., an exact representation by arbitrary precision rationals as e.g. provided by the GMP [24] library. Instead, we rely on Computational Geometry Algorithms Library (CGAL) [4], more precisely on `CGAL::Exact_predicates_exact_constructions_kernel`, which provides lazy constructions: Each coordinate is initially represented by two doubles that encode an interval containing the actual coordinate. This suffices for many decisions, for instance, a comparison with another coordinate. However, in cases in which intervals overlap, the exact coordinates are computed with GMP. Compared to the pure exact approach this usually yields a significant advantage regarding speed and memory [40].

8 Conclusion

We present a discretization of polynomial size for the continuous 1.5D Terrain Guarding Problem (TGP). This settles two open questions: (1) The continuous TGP is a member of NP and, since NP-hardness is known [33], NP-complete, and (2) it admits a Polynomial Time Approximation Scheme (PTAS), since the PTAS for the discrete TGP [23] applies to our discretization. Furthermore, we propose an algorithm for finding optimal solutions for the TGP; our implementation solves instances with up to 10^6 vertices within minutes. A key success factor are filtering techniques reducing the size of the discretization and the geometric overhead, essentially removing the computational barrier between the continuous and the discrete TGP.

References

- [1] Boaz Ben-Moshe, Matthew J. Katz, and Joseph S. B. Mitchell. A constant-factor approximation algorithm for optimal 1.5D terrain guarding. *SIAM Journal on Computing*, 36(6):1631–1647, 2007.
- [2] boost C++ libraries. <http://www.boost.org/>.
- [3] Francisc Bungiu, Michael Hemmer, John Hershberger, Kan Huang, and Alexander Kröller. Efficient computation of visibility polygons. *CoRR*, abs/1403.3905, 2014.
- [4] CGAL (Computational Geometry Algorithms Library). <http://www.cgal.org/>.
- [5] Vasek Chvátal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory, Series B*, 18(1):39–41, 1975.
- [6] Kyung-Yong Chwa, Byung-Cheol Jo, Christian Knauer, Esther Moet, René van Oostum, and Chan-Su Shin. Guarding art galleries by guarding witnesses. *International Journal of Computational Geometry and Applications*, 16(2-3):205–226, 2006.
- [7] Kenneth L. Clarkson and Kasturi R. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43–58, 2007.

- [8] Marcelo C. Couto, Pedro J. de Rezende, and Cid C. de Souza. An exact algorithm for minimizing vertex guards on art galleries. *International Transactions in Operational Research*, 18(4):425–448, 2011.
- [9] IBM ILOG CPLEX Optimization Studio. <http://www.ibm.com/software/integration/optimization/cplex-optimizer/>.
- [10] Pedro J. de Rezende, Cid C. de Souza, Stephan Friedrichs, Michael Hemmer, Alexander Kröller, and Davi C. Tozoni. Engineering art galleries. *CoRR*, abs/1410.8720, 2014. To appear.
- [11] Stephane Durocher, Pak Ching Li, and Saeed Mehrabi. Guarding orthogonal terrains. In *Proceedings of the 27th Canadian Conference on Computational Geometry*, 2015.
- [12] Alon Efrat and Sarel Har-Peled. Guarding galleries and terrains. *Information Processing Letters*, 100(6):238–245, 2006.
- [13] Stephan Eidenbenz. Approximation algorithms for terrain guarding. *Information Processing Letters*, 82(2):99–105, 2002.
- [14] Stephan Eidenbenz, Christoph Stamm, and Peter Widmayer. Inapproximability results for guarding polygons and terrains. *Algorithmica*, 31(1):79–113, 2001.
- [15] Khaled M. Elbassioni, Erik Krohn, Domagoj Matijevic, Julián Mestre, and Domagoj Severdija. Improved approximations for guarding 1.5-dimensional terrains. *Algorithmica*, 60(2):451–463, 2011.
- [16] Maximilian Ernestus, Stephan Friedrichs, Michael Hemmer, Jan Kokemüller, Alexander Kröller, Mahdi Moeini, and Christiane Schmidt. Algorithms for art gallery illumination. *CoRR*, abs/1410.5952, 2014.
- [17] Sándor P. Fekete, Stephan Friedrichs, Alexander Kröller, and Christiane Schmidt. Facets for art gallery problems. *Algorithmica*, 73(2):411–440, 2015.
- [18] Steve Fisk. A short proof of Chvátal’s watchman theorem. *Journal of Combinatorial Theory, Series B*, 24(3):374, 1978.
- [19] Stephan Friedrichs, Michael Hemmer, and Christiane Schmidt. A PTAS for the continuous 1.5d terrain guarding problem. In *Proceedings of the 26th Canadian Conference on Computational Geometry*, 2014.
- [20] Stephan Friedrichs, Michael Hemmer, and Christiane Schmidt. Exact solutions for the continuous terrain guarding problem. In *31st European Workshop on Computational Geometry*, pages 212–215, 2015.
- [21] Stephan Friedrichs, Michael Hemmer, and Christiane Schmidt. Terrain guarding problem instance library. <http://resources.mpi-inf.mpg.de/tgp/index.html#instances>, August 2015. Version 2015-08-06.
- [22] The GNU compiler collection. <http://gcc.gnu.org/>.

- [23] Matt Gibson, Gaurav Kanade, Erik Krohn, and Kasturi R. Varadarajan. Guarding terrains via local search. *Journal of Computational Geometry*, 5(1):168–178, 2014.
- [24] The GNU multiple precision arithmetic library. <https://gmplib.org/>.
- [25] Andreas Haas and Michael Hemmer. Efficient algorithms and implementations for visibility in 1.5d terrains. In *31st European Workshop on Computational Geometry*, pages 216–219, 2015.
- [26] Ferran Hurtado, Maarten Löffler, Inês Matos, Vera Sacristán, Maria Saumell, Rodrigo I. Silveira, and Frank Staals. Terrain visibility with multiple viewpoints. *International Journal of Computational Geometry & Applications*, 24(4):275–306, 2014.
- [27] Jeff Kahn, Maria Klawe, and Daniel Kleitman. Traditional art galleries require fewer watchmen. *SIAM Journal on Algebraic and Discrete Methods*, 4(2):194–206, 1983.
- [28] Matthew J. Katz and Gabriel S. Roisman. On guarding the vertices of rectilinear domains. *Computational Geometry Theory and Applications*, 39(3):219–228, 2008.
- [29] Farnoosh Khodakarami, Farzad Didehvar, and Ali Mohades. A fixed-parameter algorithm for guarding 1.5d terrains. *Theoretical Computer Science*, 595:130–142, 2015.
- [30] James King. Errata on “a 4-approximation for guarding 1.5-dimensional terrains”. http://www.cs.mcgill.ca/~jking/papers/4apx_latin.pdf. Visited 2015-08-20.
- [31] James King. A 4-approximation algorithm for guarding 1.5-dimensional terrains. In *7th Latin American Theoretical Informatics Symposium (LATIN)*, pages 629–640, 2006.
- [32] James King. *Guarding Problems and Geometric Split Trees*. PhD thesis, McGill University, 2010.
- [33] James King and Erik Krohn. Terrain guarding is NP-hard. *SIAM Journal on Computing*, 40(5):1316–1339, 2011.
- [34] Alexander Kröller, Tobias Baumgartner, Sándor P. Fekete, and Christiane Schmidt. Exact solutions and bounds for general art gallery problems. *ACM Journal of Experimental Algorithmics*, 17(1), 2012.
- [35] Aldo Laurentini. Guarding the walls of an art gallery. *The Visual Computer*, 15(6):265–278, 1999.
- [36] Der-Tsai Lee and Arthur K. Lin. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory*, 32(2):276–282, 1986.
- [37] Goran Martinović, Domagoj Matijević, and Domagoj Ševerdija. Efficient parallel implementations of approximation algorithms for guarding 1.5D terrains. *Croatian Operational Research Review*, 6(1):79–89, 2015.
- [38] Joseph O’Rourke. *Art Gallery Theorems and Algorithms*. International Series of Monographs on Computer Science. Oxford University Press, New York, 1987.

- [39] Joseph O'Rourke and Kenneth J. Supowit. Some NP-hard polygon decomposition problems. *IEEE Transactions on Information Theory*, 29(2):181–189, 1983.
- [40] Sylvain Pion and Andreas Fabri. A generic lazy evaluation scheme for exact geometric computations. *Science of Computer Programming*, 76(4):307–323, 2011.
- [41] Dietmar Schuchardt and Hans-Dietrich Hecker. Two np-hard art-gallery problems for ortho-polygons. *Mathematical Logic Quarterly*, 41:261–267, 1995.
- [42] Thomas C. Shermer. Recent results in art galleries. In *Proceedings of the IEEE*, volume 80, pages 1384–1399, 1992.
- [43] simple-svg. <http://code.google.com/p/simple-svg/>.